Readme.rtf for code related to RivWidth v. 0.4, release date February 21st, 2013.

RivWidth: A Software Tool for the Calculation of River Width from Raster-based Imagery

Corresponding Author:
Tamlin M. Pavelsky
pavelsky@unc.edu
ph: 919-962-4239
UNC Department of Geological Sciences
104 South Rd CB #3315
Chapel Hill, NC 27599

Contributing Authors:
Laurence C. Smith
UCLA Department of Geography
lsmith@geog.ucla.edu

Michael Durand
The Ohio State University

Doug Alsdorf
The Ohio State University

Updates in Rivwidth v.0.4:

1. The principal update from version three is that RivWidth now computes its own starting and ending points. As a result, there is no need to open the initial centerline in a program such as ENVI to manually identify these points. All you have to do is provide RivWidth with a binary water mask (see below), and it will compute centerlines and widths without any other input required.
2. RivWidth now computes widths for all connected channels within an image, not just for a single channel at a time. So for example if an image of the entire Mississippi Basin were provided, so long as all of the channels were connected to each other RivWidth will compute widths for all channels.
3. There have been a number of minor bug fixes/improvements in processing speed and, especially, accuracy. These especially relate to the accurate computation of orthogonals to the centerline.

Inputs required by Rivwidth:

One initial input file is required by RivWidth: a binary water mask in which water is set to 1 and nonwater is set to 0. It is permissible to leave small lakes or other water bodies not connected to the river(s) in the mask, as these will be removed during processing. The one caveat is that at present rivwidth automatically determines the width of the largest continuous water object in the image. So if there is a very large lake, you may need to remove it manually. Alternatively, you can make modifications to the code described on lines 90-92 of the code. From this initial water mask, RivWidth calculates the following (which are referred to repeatedly in Pavelsky and Smith (2008) and which were required inputs up until v0.3):

--Channel Mask: a binary mask, derived from a satellite image that assigns water pixels a value of 1 and nonwater pixels a value of 0 (TestOhio_Channel_1b). This is referred to as a channel mask, as it retains all of a river's various channels and islands.

--River mask: a binary mask that differentiates the areas within the river boundary (including islands) from those outside (TestOhio).

The first section of RivWidth now calculates the channel mask and river mask from the water mask and outputs them to files.

Outputs from Rivwidth:

Rivwidth produces two final outputs, plus some intermediate products designed to allow only part of the code to be run at any given time.

Intermediate outputs:

--River Mask and Channel Mask (described above)

--Initial Centerline: an image showing the initial rough centerline. This should be opened in IDL or ENVI and the pixel coordinates of the starting and ending pixels for the final centerline should be determined and added to the parameter file.

Final outputs:

--Width Image: an image showing the total flow width at each pixel in the centerline.

--Width spreadsheet: a .csv file containing the coordinates of each centerline pixel along with the river flow width at that pixel.

The filenames of these five files should be included in the parameter file passed to RivWidth, along with other requirements described below

Other Required Inputs to the Parameter File:

Rivwidth also needs to know the x/y dimensions of the input masks, along with the spatial resolution of the masks (in whatever units are desired, e.g. 30m for Landsat).

You will need to choose a value for "braidnarrowflag" of either 0 or 1. In general, if you are dealing with rivers that have channels only one or two pixels wide in many places, choose 1. For rivers consistently 3 pixels wide or wider, choose 0.

Finally, you can now provide information on the UTM coordinates of the upper left-hand corner of your image, and RivWidth will automatically provide the correct UTM coordinates for each centerline pixel in the final output file. If you do not have spatial reference information for your image or it is in a different projection, just leave these values as zero and ignore the relevant columns in the output file. We're hoping to add support for additional coordinate systems in the future.

Also, it is possible to set up the parameter file to turn on and off the portions of the Rivwidth code that calculate the channel and river masks, the initial centerline, and the final width values. This can be useful if, for example, you've already calculated a centerline for a given river reach and you simply want to calculate widths for a given channel mask along that centerline.

How to Run RivWidth:

1. Open up the code in IDL and compile it.
2. Open up the sample parameter file in the text editor of your choice (not a word processor like MS Word, which can add unwanted hidden characters).
3. Edit the parameter file, making sure all input and output file locations make sense. Input the correct values for the width, height and resolution of the image, UTM corner coordinates and choose a value for braidnarrow flag of 0 or 1 (see above).
4. Run RivWidth, and your final output should be available

Advanced Options:

If you feel like the output you're getting isn't correct and you want to mess around with some hidden parameters in RivWidth, here are the locations and meanings of some of them:

--At the moment, we do not consider portions of the image within 12 pixels of the image boundaries because RivWidth produces spurious measurements in these areas. If you want to change this, you can edit lines 217-220.

--If you want to work with very short channels (less than 333 pixels in length), then you will need to edit the first numeric value on line 255. This portion of the code is designed to automatically extract centerlines, and at the moment it ignores very short channels. However, the code has been tested with a range of values, and it seems to work well regardless of what value is chose. Also, if you're working in on an image with a very large number of channels, you can increase the value of <counter> at which the while loop on line 255 is shut off. It is currently set to work on 200 different centerline segments.

--If you want to compute average widths for reaches of consistent length, you can change the value of agval on line 196. I haven't tested this in a while, but I think it should still work. The value you provide is the number of pixel lengths that each averaged reach will contain. RivWidth takes into account the fact that diagonally-connected pixels are sqrt(2) pixel lengths apart, rather than 1.

--RivWidth runs much faster if it's working on relatively short centerline segments, so it automatically breaks up really long ones. At the moment, it divides anything longer than 3000 pixels into multiple reaches. This value can be changed on lines 305-307. I don't recommend going too short on this, though, because otherwise RivWidth can have problems computing correct orthogonals.

--RivWidth computes an initial centerline by taking the laplacian of a distance map. Ideally, the resulting output has centerline pixel values close to 0 and all other river pixel values close to 1. A threshold value is chosen (currently 0.8) to divide preliminary centerline pixels from other pixels. However, this value is arbitrarily chosen. Increasing the value will increase the number of centerline pixels, which will ensure more connected centerlines but may result in inaccuracies. Decreasing the value will increase the accuracy of centerlines but will also result in more gaps. You can modify this parameter on lines 463 and 464 as you choose, however.

--In calculating the orthogonal to a river, we need to choose some length of centerline against which to determine the orthogonal distance. However, this is a fairly arbitrary choice. The current segment length is 11 pixels, but it can be modified on line 688. We obtained this value through direct experimentation in the Mississippi basin, and it seems to fit the bet of all values we tested (7 to 21 pixels).

--